AMENDMENTS TO THE CLAIMS

Please amend the claims as follows:

1. (Currently Amended) A method for iterating in a dynamically typed programming language comprising:

   defining a class, wherein said class defines a special operator;

   instantiating ~~providing~~ an instance of [[a]] said class; and

   calling a special operator of said class when an operator corresponding to the special operator is called and the instance of the class is an argument of said operator,

   wherein ~~the~~ said class is written in a dynamically typed language,

   wherein said class defines at least one action to perform when said special operator is called.

2. (Original) The method of claim 1 wherein said special operator return a list of values.

3. (Original) The method of claim 2 further comprising:
   iterating through said list of values.

4. (Original) The method of claim 3 wherein said special operator is an increment operator.

5. (Original) The method of claim 1 wherein said special operator is an increment operator.

6. (Original) The method of claim 1 wherein said special operator is a decrement operator.

7. (Cancelled)

8. (Cancelled)

9. (Cancelled)

10. (Cancelled)

11. (Currently Amended) The dynamically typed programmed language iteration system of claim 1 [[7]] wherein said special operator is an increment operator.

12. (Currently Amended) The dynamically typed programming language iteration system of claim 1 [[7]] wherein said special operator is a decrement operator.

13. (Currently Amended) A computer program product comprising:

a computer usable medium having computer readable program code embodied therein configured for iterating in a dynamically typed programming language, comprising:

computer readable code configured to cause a computer to define a class, wherein said class defines a special operator;

computer readable code configured to cause a computer to instantiate ~~provide~~ an instance of [[a]] said class; and

computer readable code configured to cause a computer to call [[a]] said special operator of said class when an operator corresponding to said special operator is called and the instance of said class is an argument of the operator,

wherein the class is written in a dynamically typed language,

wherein said class defines at least one action to perform when said special operator is called.

14. (Original) The computer program product of claim 13 wherein said special operator return a list of values.

15. (Original) The computer program product of claim 14 further comprising:

computer readable code configured to cause a computer to iterate through said list of values.

16. (Original) The computer program product of claim 15 wherein said special operator is a foreach operator.

17. (Original) The computer program product of claim 13 wherein said special operator is an increment operator.

18. (Original) The computer program product of claim 13 wherein said special operator is a decrement operator.

Rejection(s) under 35 U.S.C § 103

Claims 1-6 and 11-18 stand rejected under 35 U.S.C. § 103 as anticipated by "C++ Standard Library: A tutorial and Reference" by Josuttis (hereafter "Josuttis") in view of "Learning Perl," by Christiansen (hereafter "Christiansen"). To the extent that the claims apply to the amended claims the rejection is respectfully traversed.

The invention is directed to a dynamic programming language that allows operators (*e.g.*, foreach, increment, decrement, etc.) to be applied to classes (or more specifically, instances of classes). The following is an example of the operation of the invention as recited in the amended independent claims. Initially, the user defines a class such as the following class:

```
Class X {
     var value =[1,2,3,5,7]
     operator foreach() { return value;}
}
```

The user then proceeds to write a program in a dynamically typed language that uses class X. The program may include the following piece of code:

```
var x = new X ()
foreach i x {
     y=i
}
```

Without the benefit of the present invention, the execution of foreach i x {...} would cause the program to crash. However, using the constructs of the present invention, when the line foreach i x {...} is encountered, the special operator corresponding to the foreach operator is called, *i.e.*, the foreach operator defined in class X is called. In this example, the execution of foreach operator in class X returns [1,2,3,5,7]. Once [1,2,3,5,7] is received, the original foreach operator (*i.e.*, the foreach operator in the program as opposed to the foreach operator in class X) executes the actions defined in the body of the original foreach operator (*i.e.*, y=i). The Applicant notes